

# Contents

<b>Structural Defects Over Task Errors: A Unified Framework for Diagnosing Unreliable Software Artifacts Across Agentic, Supply-Chain, and Decompilation Pipelines</b>	<b>1</b>
Abstract . . . . .	1
Introduction . . . . .	2
Background . . . . .	2
Synthesis . . . . .	3
Discussion . . . . .	6
Limitations . . . . .	7
Conclusion . . . . .	8

## Structural Defects Over Task Errors: A Unified Framework for Diagnosing Unreliable Software Artifacts Across Agentic, Supply-Chain, and Decompilation Pipelines

*Saluca Agentic AI Research Team — Saluca LLC. AI-drafted synthesis from recent arXiv preprints; for human review, not peer-reviewed.*

---

### Abstract

A recurring pattern across recent software engineering research is that the most consequential failures in complex software pipelines are not task-level errors — wrong answers, incorrect transformations, or bad predictions — but structural defects: gaps in component coverage, mismatches between artifact representations, and integration failures that mask the signal task-level monitors are designed to detect. This paper synthesizes five corpus findings spanning agentic system monitoring, software bill of materials (SBOM) generation, binary decompilation evaluation, credential leakage detection, and multi-agent LLM collaboration topology to argue a single defensible thesis: **the dominant failure mode in modern software pipelines is structural ambiguity, not functional error, and current tooling is systematically blind to this distinction.**

The evidence base draws from cs.SE preprints covering monitoring methodology [arXiv:2606.02494](#), SBOM component inclusion [arXiv:2606.02442](#), decompilation reusability metrics [arXiv:2605.29490](#), credential classification [arXiv:2605.31520](#), and multi-agent topology experiments [arXiv:2606.01490](#). Across these domains, a consistent pattern emerges: pipelines optimized for one quality axis (readability, task accuracy, alert precision) systematically underperform on orthogonal axes (functionality, structural coverage, integration completeness), and this orthogonality is rarely measured.

The primary falsification path is empirical: if a tool or monitoring regime can be shown to simultaneously maximize performance on all quality dimensions without explicit cross-axis measurement, the thesis fails. Current evidence suggests this has not been demonstrated for any of the reviewed systems. Secondary falsification requires showing that structural monitors add no predictive power beyond task-level monitors in a controlled regression study. The paper concludes with a proposal for a maturity-staged, multi-axis evaluation discipline applicable across these domains.

## Introduction

Software engineering research has long distinguished between correctness (does the program do what it should?) and reliability (does it consistently do so under realistic conditions?). What the recent corpus reveals is a third, underexamined dimension: **structural completeness** — whether the artifact, pipeline, or tool even covers the space it claims to cover.

Consider three surface-distinct problems. First, an agentic document-processing system that injects controlled task-level errors finds those errors statistically indistinguishable from clean baselines — not because the errors are small, but because structural defects in the pipeline produce variance so large that task-level signal is swamped [arXiv:2606.02494](#). Second, SBOM generation tools across five popular implementations fail to agree on which components belong in a software bill of materials, producing “ambiguity and blind spots in component inclusion” that no individual tool resolves [arXiv:2606.02442](#). Third, binary decompilation evaluation reveals a “reusability cliff” where the best decompiler-LLM pair reaches 22.3% program-level behavioral overlap but only 1.2% exact output match — a 20-point gap that syntactic similarity metrics would entirely miss [arXiv:2605.29490](#).

These are not coincidental. They share a common structure: a pipeline produces an artifact, the artifact is evaluated on a single axis, and the single-axis evaluation is systematically optimistic about dimensions that were never measured. The motivating question for this paper is: **what is the generalizable structure of this failure pattern, and what evaluation discipline would make it visible?**

The answer, we argue, requires decomposing pipeline evaluation into orthogonal quality dimensions, instrumenting at multiple scopes (within-run, cross-run, structural), and treating variance — not mean performance — as the primary diagnostic signal. This is not a claim about any specific tool being bad; it is a claim about the evaluation frameworks those tools are embedded in being incomplete.

---

## Background

### 2.1 The Single-Axis Evaluation Trap

Evaluation frameworks in software engineering tend to crystallize around the metrics that are easiest to compute. For decompilation, syntactic similarity (BLEU scores, edit distance) is tractable and well-understood; behavioral equivalence requires executing both the original and decompiled code under matched inputs, which is expensive [arXiv:2605.29490](#). For SBOM generation, listing which components a tool *does* identify is straightforward; characterizing the space of components it *should* identify requires a ground-truth taxonomy of Component Inclusion Mechanisms (CIMs) that did not previously exist [arXiv:2606.02442](#). For credential detection, binary classification (secret vs. not-secret) is simpler to implement than three-class classification that distinguishes genuine credentials from placeholders — and the simpler formulation produces the high false-positive rates that make existing tools operationally unreliable [arXiv:2605.31520](#).

The pattern is consistent: the evaluation axis that is easiest to measure is adopted, and the axes that require additional instrumentation are deferred or ignored. The consequence is that tools optimized against the measured axis may actively underperform on unmeasured axes — a phenomenon the decompilation study makes explicit by showing that compiler settings maximizing readability minimize functionality, and vice versa [arXiv:2605.29490](#).

## 2.2 Structural vs. Task-Level Failure Modes

The monitoring literature distinguishes two failure regimes that are operationally distinct but often conflated. Task-level failures are errors in the output of a specific operation: a wrong classification, an incorrect transformation, a missed credential. Structural failures are defects in the pipeline itself: an integration gap between stages, a component that is never invoked, a coverage hole in the artifact representation.

The agentic monitoring study formalizes this distinction with a concrete measurement: within-run monitors surface deterministic stage defects with coefficient of variation  $CV = 0.02$ , cross-run monitors surface stochastic integration consequences with  $CV = 1.25$ , and injected task-level errors produce  $CV$  indistinguishable from clean baselines [arXiv:2606.02494](#). This is not a subtle statistical effect — the structural monitor identifies an integration gap with  $CV = 0.00$  (perfect consistency), while task-level errors are invisible at the same monitoring scope. The implication is that deploying only task-level monitors in a structurally defective pipeline produces confident but meaningless evaluations.

## 2.3 Artifact Lifecycle and Governance

A third background concept relevant to this synthesis is the distinction between artifacts as outputs and artifacts as persistent operational entities. The conventional model treats a pipeline artifact (a compiled binary, a generated SBOM, an agent-produced document) as a point-in-time output evaluated at delivery. An emerging alternative, formalized in the governed runtime evolution literature, treats artifacts as persistent runtime capabilities subject to lifecycle management, validation, and rollback [arXiv:2605.27328](#).

This distinction matters for evaluation because point-in-time evaluation captures only the artifact’s state at one moment in a potentially evolving lifecycle. If artifacts are revised, mutated, or reused across runs — as the HarnessMutation framework proposes — then evaluation must track not just quality at delivery but quality drift over the artifact’s operational lifetime. The governance constraints proposed (explicit validation, traceability, rollback) are precisely the mechanisms that would make multi-axis evaluation tractable across an artifact’s lifecycle [arXiv:2605.27328](#).

---

## Synthesis

### 3.1 Variance as a Diagnostic Signal, Not a Noise Floor

**Claim:** Coefficient of variation ( $CV$ ) across repeated runs is a more informative diagnostic signal than mean performance for identifying structural defects, because structural defects produce characteristic variance signatures that task-level metrics cannot replicate.

**Evidence:** The agentic monitoring study directly measures this. Within-run monitors on deterministic stage defects produce  $CV = 0.02$  — low variance, high consistency, indicating a fixed structural property. Cross-run monitors on stochastic integration consequences produce  $CV = 1.25$  — high variance, indicating a non-deterministic integration boundary. Critically, injected task-level errors produce  $CV$  indistinguishable from the clean baseline, confirming that structural defects mask task-level signal rather than amplifying it [arXiv:2606.02494](#). The structural monitor achieves  $CV = 0.00$ , identifying an integration gap with perfect consistency across 220 runs.

**Caveat:** This finding is reported on a synthetic testbed with controlled error injection across 120 document bundles. The CV signatures observed may be specific to the document-processing domain or to the particular structural defects injected. Whether  $CV = 0.00$  for structural monitors generalizes to other pipeline architectures is untested.

**Falsification path:** Run the same monitoring regime on a pipeline where structural defects are stochastic (e.g., a probabilistic integration boundary). If CV for the structural monitor rises above the task-level CV in that setting, the claim that  $CV = 0.00$  is a reliable structural signature fails. Alternatively, construct a task-level error that produces  $CV > 1.0$  in a structurally sound pipeline; if such errors exist, the claim that high CV is diagnostic of structural (not task-level) failure is falsified.

### 3.2 Component Coverage Gaps Are Systematic, Not Random

**Claim:** Across SBOM generation tools, coverage gaps in component inclusion are not random sampling failures but systematic blind spots tied to which Component Inclusion Mechanisms (CIMs) each tool’s architecture recognizes — and no current tool covers all CIMs.

**Evidence:** The SBOM analysis systematically evaluates four popular tools (cdxgen, syft, trivy, ORT) plus the Microsoft sbom-tool against a ground-truth CIM taxonomy across six programming languages (Python, Java, Go, PHP, Rust, C). The finding is not that tools perform poorly on average, but that “no tool covers all identified CIMs and that common gaps exist across tools” [arXiv:2606.02442](https://arxiv.org/abs/2606.02442). The gaps are architectural: tools built around package-manager manifests miss dynamically linked components; tools built around binary scanning miss source-level dependencies. These are not random misses — they are structural blind spots determined by the tool’s design assumptions about what a “component” is.

**Caveat:** The ground-truth CIM taxonomy is itself a construct of this study. If the taxonomy is incomplete or if reasonable alternative taxonomies would classify some “gaps” as intentional scope exclusions, the severity of the finding changes. The study does not report inter-rater reliability for the ground-truth classification.

**Falsification path:** Construct a CIM taxonomy through a process with documented inter-rater agreement, then re-evaluate the same tools. If tool coverage gaps disappear or become random under an alternative taxonomy, the claim that gaps are systematic and architectural is weakened. If gaps persist across multiple independent taxonomies, the claim is strengthened.

### 3.3 Quality Axes Are Orthogonal, Not Correlated — And Optimization Along One Axis Degrades Others

**Claim:** In complex software transformation pipelines, quality dimensions (readability, recompile-ability, functionality; or precision, recall, false-positive rate) are not correlated proxies for a single underlying quality but genuinely orthogonal axes where optimization along one axis actively degrades performance on others.

**Evidence:** The decompilation study makes this explicit with two concrete findings. First, “-O3 yields the lowest readability but the highest functionality, and Clang gives lower readability than GCC but 2.6x higher functionality” [arXiv:2605.29490](https://arxiv.org/abs/2605.29490). These are not marginal differences — 2.6x is a large effect. Second, “cross-decompiler variation at the functional level is 20x, far larger than the 1.6x cross-LLM variation,” showing that the choice of decompiler (a structural choice about

the pipeline architecture) dominates the choice of repair LLM (a task-level choice) by an order of magnitude.

The credential detection study provides a parallel finding: binary classification (the simpler axis) produces high false-positive rates that make tools operationally unreliable. Adding a third class (placeholder/weak credentials) improves the operationally relevant metric (reducing high-severity alerts by 33%, from 373 to 250) without sacrificing recall on genuine leaks [arXiv:2605.31520](#). The two-class framing was not just simpler — it was actively optimizing the wrong axis.

**Caveat:** Both studies are evaluated on specific datasets (240 atomic test functions for decompilation; 9,426 samples for credential detection). The orthogonality finding may be dataset-specific. In particular, if the test functions in the decompilation benchmark are systematically selected to maximize the readability-functionality tradeoff, the 2.6x effect may overstate the general case.

**Falsification path:** Identify a pipeline setting that simultaneously maximizes readability and functionality in the decompilation benchmark. If such a setting exists and was not evaluated, the orthogonality claim is weakened. For credential detection, show that a binary classifier can match the three-class model’s 33% alert reduction without the explicit placeholder class; if achievable through threshold tuning alone, the structural necessity of the third class is in question.

### 3.4 Topology Determines Output Quality More Than Model Capability in Multi-Agent Systems

**Claim:** In multi-agent LLM collaboration for software design, the structural topology of agent interaction (who reviews whom, in what order, with what authority) explains more variance in output quality than the capability of individual models, and some topologies are categorically broken regardless of model quality.

**Evidence:** The controlled experiment across 12 collaboration topologies and 520 runs finds that “parallel merge is fundamentally broken — all three evaluators place merge variants in the bottom tier (3.65-3.79), due to token starvation and the Frankenstein effect” [arXiv:2606.01490](#). This is a categorical finding, not a marginal one: the bottom-tier score (3.65-3.79 out of 5.0) versus the top-tier score (4.637) represents a gap that persists across all three independent evaluators (GPT-OSS 120B, Claude Opus 4.6, Claude Sonnet 4.6). The structural adversarial topology (v4b) and cross-model review topology rank first and second by all evaluators, while merge variants rank last by all evaluators — the topology ranking is more consistent across evaluators than the model-specific quality rankings.

The evaluator disagreement finding reinforces this: evaluators “agree v4b is best and v3 is worst, but disagree sharply on v2b (Claude d=1.44 vs. GPT-OSS d=0.45)” [arXiv:2606.01490](#). The disagreement is on intermediate topologies, not on the structural extremes. This suggests that topology determines the ceiling and floor of achievable quality, while model-specific factors determine position within that range.

**Caveat:** The experiment uses automated evaluators (three LLM instances) rather than human expert evaluation. If the evaluator models have systematic biases toward certain output styles (e.g., preferring adversarially generated content), the topology rankings may reflect evaluator bias rather than genuine design quality. The study reports cross-validation but does not report human ground-truth validation.

**Falsification path:** Replicate the experiment with human expert evaluators on a stratified sample

of outputs. If human evaluators reverse the topology rankings — placing merge variants above adversarial variants — the claim that topology determines quality ceiling is falsified. If human evaluators agree with the automated rankings on the structural extremes (v4b best, merge worst) but disagree on intermediates, the claim survives in its weaker form.

### 3.5 Structural Priors Outperform End-to-End Learning for Topology-Critical Tasks

**Claim:** For tasks where the correct output depends on preserving topological structure (graph connectivity, component relationships, integration boundaries), injecting a deterministically extracted structural prior into a learned model’s input outperforms end-to-end learning without that prior — and the improvement is concentrated at the structural level, not the content level.

**Evidence:** The EdgeFlow study evaluates VLM-based flowchart conversion with and without a Canny edge map as a structural prior. The improvement from adding the edge map is: node-level F1 +17.39 percentage points, edge-level F1 +16.94 percentage points, path-level F1 +11.06 percentage points [arXiv:2605.27332](#). Critically, this improvement requires no annotated training data and no domain-specific fine-tuning — the structural prior is injected at inference time as an additional input channel. The improvement is training-free, which means it cannot be attributed to additional training signal; it must be attributed to the structural information itself.

The cross-dataset finding is equally important: “cross-dataset evaluation results on a public synthetic benchmark show no significant improvement” [arXiv:2605.27332](#). The structural prior helps on industrial data (where topology is genuinely complex and variable) but not on synthetic data (where topology may be simpler or more regular). This suggests the benefit of the structural prior scales with the structural complexity of the task, not with dataset size.

**Caveat:** The comparison is between EdgeFlow (VLM + edge map) and off-the-shelf VLMs without the edge map. It does not compare against a fine-tuned VLM trained on flowchart data, which might learn equivalent structural priors from training examples. The 17-point improvement may partly reflect the baseline VLM’s lack of domain adaptation rather than the intrinsic value of the structural prior.

**Falsification path:** Fine-tune a VLM on the IndusReqFlow training set without the edge map. If the fine-tuned model matches EdgeFlow’s node-level and edge-level F1 without the structural prior, the claim that the prior provides irreducible benefit is falsified. If fine-tuning cannot close the gap, the claim that structural priors provide information not learnable from data is strengthened.

---

## Discussion

### What This Implies

The synthesis across these five findings points toward a discipline that might be called **multi-axis structural evaluation**: the practice of explicitly measuring pipeline quality along orthogonal dimensions, instrumenting at multiple scopes, and treating variance signatures as diagnostic signals for structural (not just task-level) defects.

The practical implication for SBOM tooling is that coverage gap analysis requires a CIM taxonomy developed independently of the tools being evaluated — current tools cannot self-report their own blind spots [arXiv:2606.02442](#). The practical implication for decompilation benchmarking is that any benchmark reporting only readability or only recompilability is providing information about at



most one-third of the relevant quality space [arXiv:2605.29490](#). The practical implication for agentic system deployment is that task-level monitoring should be understood as a complement to, not a substitute for, structural monitoring — and that deploying task-level monitors before structural monitors are in place may produce false confidence [arXiv:2606.02494](#).

The topology findings from multi-agent collaboration [arXiv:2606.01490](#) and flowchart conversion [arXiv:2605.27332](#) suggest a convergent principle: structural choices (how agents are connected, what priors are injected) dominate capability choices (which model is used, how much training data is available) for tasks where topology is load-bearing. This is a testable hypothesis, not an assertion — the falsification paths above are concrete.

The governed runtime evolution framework [arXiv:2605.27328](#) suggests that multi-axis evaluation should extend across the artifact lifecycle, not just at delivery. If artifacts evolve through HarnessMutation-style adaptation, a single-point evaluation at deployment may be systematically optimistic about long-run quality. Lifecycle-aware evaluation would require tracking quality metrics across mutation events, with rollback triggered by multi-axis degradation rather than single-axis threshold violations.

## What This Does NOT Imply

This synthesis does not imply that task-level evaluation is useless. The credential detection study demonstrates that task-level metrics (recall on genuine credentials, precision on alerts) are necessary inputs to the evaluation framework — the three-class model is validated precisely because it improves on task-level metrics while adding structural discrimination [arXiv:2605.31520](#). The claim is that task-level metrics are insufficient, not that they are irrelevant.

This synthesis does not imply that structural monitors are always available or cheap to construct. The SBOM study’s ground-truth CIM taxonomy required substantial domain expertise to build [arXiv:2606.02442](#). The agentic monitoring study’s structural monitor required a testbed of 220 runs with controlled error injection [arXiv:2606.02494](#). In practice, the cost of building structural monitors may exceed the cost of the tools being evaluated — this is a real barrier that the synthesis does not resolve.

This synthesis does not imply that the findings generalize across all software domains. Each study is bounded by its dataset, its domain, and its evaluation methodology. The decompilation findings are specific to the five decompilers and three repair LLMs evaluated [arXiv:2605.29490](#). The topology findings are specific to 12 collaboration topologies and 8 design tasks [arXiv:2606.01490](#). Generalization claims require replication across domains, which none of these studies provide.

Finally, this synthesis does not endorse any specific tool, topology, or architectural choice. The structural adversarial topology ranking first in the multi-agent experiment [arXiv:2606.01490](#) does not mean adversarial prompting is universally beneficial — the finding is domain-specific to software architecture design tasks evaluated by LLM judges.

---

## Limitations

**Preprint status:** All corpus sources are arXiv preprints, not peer-reviewed publications. The findings have not been subject to external scrutiny. In particular, the multi-agent topology experiment [arXiv:2606.01490](#) relies on automated LLM evaluators rather than human ground truth, and the

agentic monitoring study [arXiv:2606.02494](#) is explicitly self-described as “Stage 1 evidence.” These are not disqualifying, but they require epistemic caution.

**Dataset scale:** Several studies operate at scales that may not support strong generalization claims. The decompilation benchmark contains 240 atomic test functions [arXiv:2605.29490](#). The credential detection dataset contains 9,426 samples [arXiv:2605.31520](#). The agentic monitoring study covers 220 runs across 120 document bundles [arXiv:2606.02494](#). These are reasonable for exploratory studies but insufficient for strong distributional claims.

**Mechanism underspecification:** The synthesis identifies a pattern (structural defects dominate task-level errors) but does not fully specify the mechanism by which structural defects mask task-level signal. The agentic monitoring study provides a variance-based account [arXiv:2606.02494](#), but whether this account generalizes to SBOM gaps or decompilation reusability cliffs requires theoretical work not present in the corpus.

**Construct validity:** The key constructs — “structural defect,” “task-level error,” “quality axis” — are operationalized differently across studies. The agentic monitoring study operationalizes structural defects as integration gaps between pipeline stages [arXiv:2606.02494](#). The SBOM study operationalizes them as CIM coverage gaps [arXiv:2606.02442](#). The decompilation study operationalizes them as the gap between readability and functionality axes [arXiv:2605.29490](#). Whether these are instances of the same underlying construct or merely analogically similar phenomena is a theoretical question this synthesis cannot resolve.

**Selection bias:** The corpus is limited to 10 papers from cs.SE, cs.PL, and cs.AR from a 30-day window. The synthesis reflects what was published in this window, not the full state of the field. Papers that would disconfirm the structural-defects-dominate thesis — for example, studies showing that task-level monitors are sufficient in practice — may exist but are not in the corpus.

**Categorical semantics scope:** The Freyd operad / multicategorical semantics paper [corpus:arxiv:2605.31389... — correction: that is Neuroforger] and the multicategorical semantics paper [arXiv:2605.21337](#) are not directly integrated into the synthesis. The categorical semantics work addresses foundational questions about untyped effectful computation that do not map cleanly onto the structural defect framework. Including them would require speculative connections not supported by the abstracts.

---

## Conclusion

Across agentic system monitoring, SBOM generation, binary decompilation evaluation, credential leakage detection, and multi-agent collaboration topology, a consistent structural pattern emerges: the failure modes that matter most are not task-level errors but structural gaps — integration defects, component coverage blind spots, and quality-axis orthogonalities — and current evaluation frameworks are systematically blind to these gaps because they instrument only the axis that is easiest to measure. The evidence from the reviewed corpus suggests that variance signatures (not mean performance) are the most reliable diagnostic signal for structural defects, that topology and structural priors dominate model capability for topology-critical tasks, and that multi-axis evaluation is necessary but currently absent from standard practice across these domains. Each of these claims carries a concrete falsification path: if controlled experiments demonstrate that task-level monitors are sufficient, that quality axes are correlated rather than orthogonal, or that



structural priors add no information beyond what fine-tuning captures, the thesis fails — and the field will be better for knowing it.

---